

SMART NODE: INTELLIGENT TRAFFIC INTERPRETATION

Nan Hu, Hamid Aghajan, and Tianshi Gao
Wireless Sensor Networks Lab
Department of Electrical Engineering
Stanford University
350 Serra Mall, Stanford, CA 94305 USA
+1 650-736-4323, nanhu@stanford.edu

and

Jaime Camhi, Chu Hee Lee, and Daniel Rosario
Volkswagen Group of America, Inc. Electronics Research Laboratory
4009 Miranda Avenue, Suite 150, Palo Alto, CA 94304 USA
+1 650-496-7037, Jaime.Camhi@vw.com

Abstract: The wide application of GPS navigation system on vehicles has increased the quality of the driving experience. The current GPS based navigation system, however, suffers from the limited real-time traffic updates, and therefore poses a need to cover more areas with traffic sensor infrastructure. In this paper, we propose a framework for intelligent traffic interpretation aiming to enhance the quality of vehicle probe data for navigation applications. In the proposed framework, traffic interpretation can be achieved by either a single smart camera or through the communication within a network of smart cameras mounted on vehicles. Each individual smart camera collects sensory data, which are further processed by different modules of the system to estimate the speed of the nearby vehicles on individual lanes. The sensory data that are collected include sequence of images from video camera, motion parameters from vehicle CAN bus, position information from GPS, and road information from digital map. The estimated speeds are then either used to infer the traffic status within the smart camera per se or shared among the network to achieve a better interpretation by aggregation. The system and the framework will be described in detail in this paper.

Key Words: *smart camera, vehicle detection, vehicle tracking, lane detection, speed estimation, traffic status inference, probe data*

INTRODUCTION

The current GPS navigation system calculates the routes based on traveling distance and the speed limit for different segments of roads. Even though some navigation systems do incorporate traffic information collected from broadcasting channels, that information is by no means real-time and up-to-date. The resulting routes are then suboptimal or even far from optimal, and hence lengthen the traveling time as well as increase air pollution and decrease the fuel efficiency. Considering the fact that the total number of vehicle miles traveled (VMT) on highways continues to increase across the U.S., with people driving greater distances and for longer periods of time, there is a strong need to make the real-time traffic information available, especially from the environmental viewpoint.

Most of the smart transportation systems nowadays have been developed under an observer-centric paradigm, operating based on a large installed set of sensing devices. A change of paradigm from the observer-centric network to a user-centric one can reveal some of the potentials of employing a network of cameras in smart transportation systems. For example, cameras mounted on vehicles can act as data source in lieu of installed roadside or embedded sensors in an area. Moreover, network established locally between the vehicles in an area can offer faster access to real-time traffic information in the area of interest. In user-centric paradigms, the system is tasked to offer services to its users by being aware of and responsive to their status, context, and the desired information they need to access. The aim of its operation would be to sense the environment, collect and compile to-the-point information and deliver it to the users in a timely manner to assist them in a variety of ways, from comfort to safety to efficiency. As part of the "Clean Air, a Viable Planet" research initiative of Audi [1], a premium brand of the Volkswagen Group, we propose an intelligent traffic interpretation framework which is a user-centric network.

There are several existing work using vision sensor network, such as [2, 3, 4, 5]. But most of them are for in-door applications. Because of the large difference in the application environment, in-door and our-door applications generally employ methodology different from each other. In our proposed framework, multi-sensors are deployed to collect out-door environmental data. The sensors deployed include a video camera, the vehicle CAN bus and a GPS unit, all of which are synchronized. Besides the sensors, a digital map database is also available to provide road information when acquired with the GPS position readings.

The proposed system is composed of four modules, namely vision module, fusion module, reasoning module, and communication module. The video camera is calibrated beforehand, and the vision module reads the position information from the GPS unit and acquires the digital map database for the total number of lanes on that segment of road. Equipped with this information, it processes the video taken by the camera and detects vehicles and lanes in the surroundings as well as estimates our own traveling lane number. The detected vehicles are then tracked and the distance and the traveling speed of each tracked vehicle relative to our own vehicle are also estimated in the vision module. These estimates are passed to the fusion module, which will further estimate the absolute speed of each detected vehicles around after combining our own traveling speed from the CAN bus. The fusion module also takes the position reading of the GPS unit and acquires the speed limit on that segment of road and pass all the information to the reasoning module. If the

communication system indicates there are smart nodes nearby, estimates from those smart nodes are also passed to the reasoning module and the traffic status on a nearby segment of road will be inferred when all the available information are considered in the reasoning module.

FRAMEWORK DESCRIPTION

The framework of traffic interpretation is composed of four modules - vision module, fusion module, reasoning module, and communication module. The data flow of the proposed framework is shown in Figure 1.

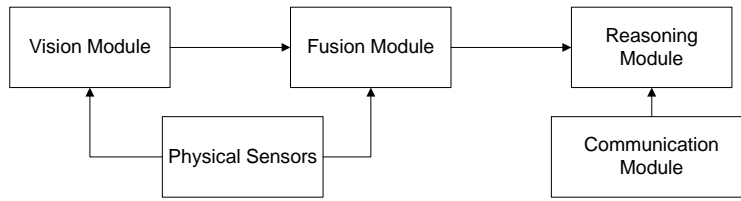


Figure 1: The data flow of the system.

The detailed description of the components and functionality of the individual modules are described below. As the way of communication is outside the scope of this paper, the communication module will not be presented, but the connection between the communication module and the reasoning module will be introduced. Before we present the different modules introduced above, a brief description of the physical sensors will first be given.

Physical Sensors

The physical sensors include a video camera, the vehicle CAN bus, a GPS unit, all of which are synchronized, together with a digital map database. The video camera is mounted at the back of the front mirror facing forward. The CAN bus is connected to both the camera and the GPS unit. The video camera is the master and synchronized with the CAN bus. Once a frame is recorded, the camera sends an interrupt signal to the CAN bus and reads from it the most recent reading of the car motion parameters and the GPS readings. The digital map is asynchronous with the system and can be read any time when acquired.

Vision Module

The whole vision module is separated into 5 tasks, vehicle detection, vehicle tracking, lane detection, relative distance/speed estimation, and vehicle/lane grouping. Figure 2 shows a flow chart of the connection among the tasks.

The vehicles around our own vehicle are detected in the vehicle detection task and the results are sent to the tracking task to track the motion of the vehicle on the image plane over time. The lanes are simultaneously detected in the lane detection task and the physical position of the lanes in the

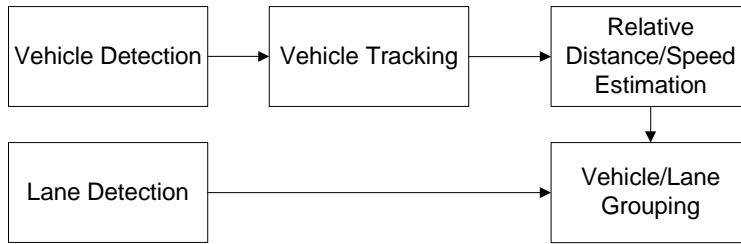


Figure 2: The connection among different tasks inside the vision module.

3-D world are estimated through the use of the camera intrinsic and extrinsic parameters. The lane number where our own vehicle is traveling is also inferred inside this task. From the vehicle tracking task, the temporal positions of the vehicles on the image plane are back projected to the ground plane in the 3-D world in the relative distance/speed estimation task, and relative speed are then estimated from the relative distance. After the position of the vehicles and the lanes in the image plane are estimated, they are merged in the vehicle/lane grouping module where vehicles are assigned to different lanes.

Vehicle Detection

The vehicle detection is done on a frame by frame basis. We trained a cascade of boosted classifier developed by Viola-Jones [6] with an extended set of haar-like features [7]. The set of haar-like filters are shown in Figure 3.

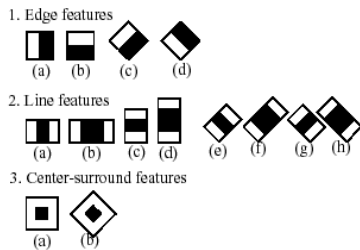


Figure 3: The set of haar-like filters used in the detector [8].

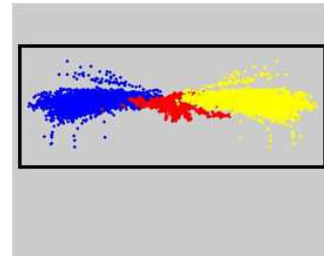


Figure 4: The ROI of the image (inside the black rectangle).

The idea of using a cascaded classifier is to reject most of the negative samples (non-vehicles in our case) as early as possible and leaves a very small subset of hard-to-differentiate samples at the end, which can dramatically increase the speed of the detector. Each cascade of the classifier is a strong classifier composed of stages of weak classifiers. Any sample that is rejected in any cascade of the classifier will be classified as negative (non-vehicle). Only a sample passes all the cascades can it be declared positive. Each weak classifier itself is not required to perform very well. As long as it performs better than random guess (has a detection rate greater than 0.5), when these weak classifiers are staged, the performance could be greatly improved.

Adaboost algorithm is employed here to build the strong classifier from the training of weak classifiers. From the training images, an overcomplete feature set is created by applying the haar-like

filters, where Adaboost algorithm performs a feature selection for weak classifiers. In each iteration, Adaboost selects the weak classifier built upon a single component of the feature set with the smallest weighted classification error. The cascades of strong classifiers are organized with increasing complexity, the reason of which is to reject negative (non-vehicle) samples as early as possible to save computational cost. Each strong classifier within the cascade is defined as

$$H(\mathbf{x}) = \begin{cases} 1, & \sum_{i=1}^N \alpha_i h_i(\mathbf{x}) \geq \frac{1}{2} \sum_{i=1}^N \alpha_i, \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

where \mathbf{x} is the feature vector of a sample, $h_i(\cdot) \in \{0, 1\}$ is the i -th weak classifier, α_i is the weight of the i -th weak classifier and N is the total number of weak classifiers in that strong classifier. As can be seen, the weak classifiers are combined to the strong classifier by a weighted majority voting. Selected positive and negative training samples are shown in Figure 5.



Figure 5: Selected positive and negative training samples.

As the mounting point of the camera is known and fixed beforehand, the region where the vehicles could appear in the images can be estimated. From the training data, a point cloud of the position of the center of the vehicles on the image is drawn, and the Region Of Interest (ROI) is set accordingly as shown in Figure 4. In this way, the detector is not required to search the whole image but only the ROI, which saves computational cost as well as decreases the total number of false alarms.

Lane Detection

The average vehicle speed on each lane is an important cue in traffic status reasoning. The lane detection task estimates the position of each lane in the 3-D world which can be further collaborate with detection and tracking tasks to assign vehicles to their own lanes. In addition, the ego-lane number (the lane that our own vehicle is traveling on) can also be inferred inside this task by classifying the road boundaries from the detected lane boundaries.

As Figure 6 shows, the lane detection task consists of three subtasks, i.e. individual lane detection, tracking/smoothing, and ego-lane number inference. The functionality for each of them are as follows:

Lane Detector: The input to the lane detector is a single frame and the output is the parameterized lane boundary candidates. The data flow chart for the lane detector is shown in Figure 7. Only the ROI of the frame is considered as possible lane area. The steerable filter [12] was employed to extract lane marker candidates out of which only those with their strength, intensity and orientation values over some predefined threshold remained in order to reduce noise. The standard Hough transform was applied upon the candidates to fit lines, and the vanishing point constraint was

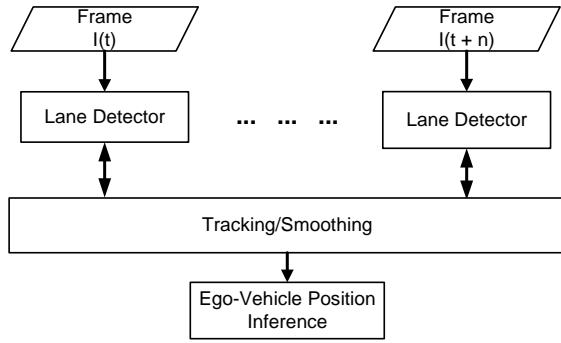


Figure 6: Lane detection task block diagram.

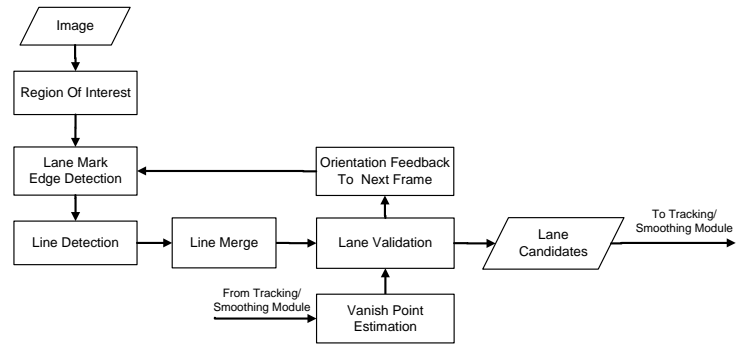


Figure 7: Flow chart for the lane detector.

then used to filter out impractical fitting. To speed up lane marker extraction, the direction of the gradient of the lane boundary in the current frame was fed back to the next frame as prior knowledge.

Tracking/smoothing: After the detection was done based on the current frame per se, these candidate lane boundary lines were passed to the tracking/smoothing subtask where information from previous time slices were combined to induce temporal consistency. In the implementation, a FIR low pass filter was applied to the lane candidates to smooth the results. To reduce false alarms, a confidence level was updated for each lane boundary over time and only when the confidence level is over some threshold can the lane boundary be declared as a detected one.

Ego-lane number inference: As normally the road boundaries (the left lane marker of the leftmost lane and the right lane marker of the rightmost lane) are drawn in solid line and the middle lane boundaries (lane markers that are not the road boundaries) are drawn in dashed or dotted line, a classification of the detected lane markers based on the continuity can tell the road boundaries from the middle lane boundaries. Ego-lane number is estimated by simply counting how many lanes we are from the detected road boundary.

Vehicle Tracking

Once the vehicle detection results are ready, the position and size of each detected object is sent to the vehicle tracker. Here, we propose a Kalman filter based greedy algorithm for tracking. Figure 14 shows a flow chart of the proposed tracking algorithm.

Within the algorithm, two lists of objects are kept in the memory, one is the transient list and the other is the confirmed list. The transient list can be deemed as a buffer for the confirmed list. It accumulates confidence for each detected object. Once the confidence goes above some threshold, the object is confirmed as an vehicle and the record is moved to the confirmed list, where tracking is continued with a template matching method. Hence, for the confirmed list, objects can only be added by being transferred from the transient list. Each object in the confirmed list has a matching score. If the matching score is below some threshold, the object is deleted from the confirmed list

and the vehicle is then thought of as having disappeared. As we are more interested in the relative position and the relative speed of the vehicle rather than the identity, we do not buffer the vehicle for reappearance once it disappeared as it only deals with the identity of the vehicle.

The detected position and size of the vehicle in each frame of the video sequence can be deemed as a measurement of the real position and size corrupted by the noise. Kalman filter provides a way to estimate the real position and size, called the state, given the noisy measurement in the current frame, as well as predict the position and size in the next frame. Here, we assume a constant speed model. Let us define $\mathbf{z} = (x, y, s, \Delta x, \Delta y, \Delta s)$ to be the state vector, where (x, y) are the 2-D position on the image plane, s is the size of the vehicle, $(\Delta x, \Delta y, \Delta s)$ are the change between adjacent frames, and $\mathbf{x} = (x_m, y_m, s_m)$ is the noisy measurement as we only observe the position and size. Figure 8 shows the graph of the model. The gray nodes represent the observations (measurements) and the white nodes are the states.

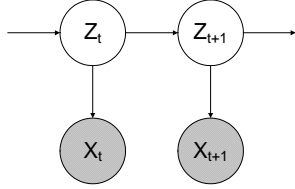


Figure 8: The graph for the Kalman filter.



Figure 9: An example of the touch point area (shown in red ellipse).

The transition of the Kalman filter satisfies

$$\mathbf{z}_{t+1} = H\mathbf{z}_t + w \quad (2)$$

$$\mathbf{x}_{t+1} = M\mathbf{z}_{t+1} + u \quad (3)$$

where in our problem, $H = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$, and $M = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$ are the

transition and measurement matrix, $w \sim \mathcal{N}(0, R)$ and $u \sim \mathcal{N}(0, Q)$ are Gaussian noises with covariance matrices R and Q .

Given prior information from past observations $\mathbf{z}_t \sim \mathcal{N}(\mu_t, \Sigma)$, the update once the measurement is available is given by

$$\mathbf{z}_t^{\text{post}} = \mu_t + \Sigma M^T (M \Sigma M^T + R)^{-1} (\mathbf{x}_t - M \mu_t) \quad (4)$$

$$\Sigma^{\text{post}} = \Sigma - M^T (M \Sigma M^T + R)^{-1} M \quad (5)$$

$$\mu_{t+1} = H \mathbf{z}_t^{\text{post}} \quad (6)$$

$$\Sigma = H \Sigma^{\text{post}} H^T + Q \quad (7)$$

As can be seen the z_t^{post} can be deemed as a correction once the measurement x_t is given, and μ_t can be thought of as a prediction. Hence, in our problem, if the tracked vehicle has a match found in the current frame, we declare a measurement is given and the position and size components of the correction z_t^{post} are output as the vehicle's position and size in the image plane, otherwise the prediction is the output and there will be no update because of the lack of measurement.

Within the two lists, confirmed object list $O^c = \{O_i^c\}$ and transient object list $O^{tr} = \{O_j^{tr}\}$, O^c has a higher priority than O^{tr} , which means for every time slice, O^c is updated prior to O^{tr} . Each object O_i^c or O_j^{tr} is associated with a confidence level, and in addition each O_i^c has an embedded template, i.e. the image of the vehicle which are updated every n_f frames of successful matching. A brief description of the tracking algorithm is shown in Algorithm 1.

The matching scheme for transient objects and the confirmed objects are not the same. For transient objects, matches are found by applying geometrical constrains from the prediction of the Kalman filter. For confirmed objects, template matching scheme is used, where a sparse optical flow method [8] is used to find correspondences between the template and the predicted area by Kalman filter, and RANDOM SAMPLE CONSENSUS (RANSAC) algorithm [9] is employed to estimate an affine transform and the position and size are transformed accordingly and deemed as the measurement.

Relative Distance/Speed Estimation

With a calibrated camera, the detected vehicles can be back projected to the 3-D world, however, with ambiguity the same as for any monocular camera system. In our system, the problem can be solved with one more constrain that the vehicles are traveling on the ground plane, which by camera calibration, the height of the camera to the ground plane can be estimated. Here we assume a simple pin-hole camera model as shown in Figure 10.

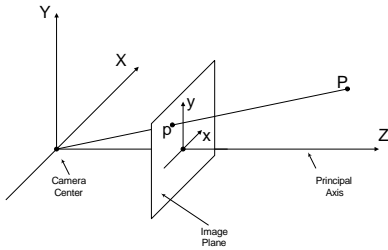


Figure 10: The pin-hole camera model.

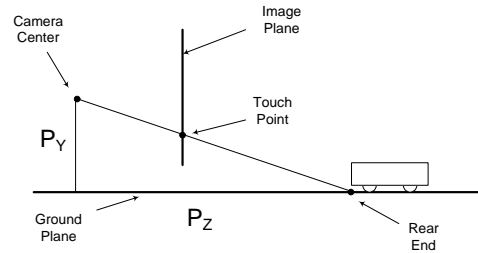


Figure 11: Relative distance estimation from the back projection of the touch point.

From the pin-hole model, for any point $P = (P_X, P_Y, P_Z)$ in the 3-D world, the image point $p = (p_x, p_y)$ is given by

$$p_x = f_x \frac{P_X}{P_Z} + o_x, \quad p_y = f_y \frac{P_Y}{P_Z} + o_y \quad (8)$$

where f_x, f_y are the focal length on x and y direction, (o_x, o_y) is the camera center. We assume the camera optical axis is parallel to the ground plane, which in experiment we estimated the angle

Algorithm 1 The vehicle tracking algorithm

Require: $O^c = \{O_i^c\}$, $O^{tr} = \{O_j^{tr}\}$, current detection $D_t = \{D_{ti}\}$, frame I_t

```
for all  $O_i^c$  in  $O^c$  do {// updating the confirmed list}
  MakePrediction(KalmanFilter)
  if FindMatch( $I_t$ ) then
    if consecutively found match in  $n_f$  frames then
      UpdateTemplate()
    end if
    DeleteSimilarObjects( $D_t$ )
    Update(KalmanFiter)
    AccumulateConfidence()
  else
    DiminishConfidence()
    if confidence < threshold  $h_1$  then
      DeleteObject( $O_i^c$ )
    end if
  end if
end for
for all  $O_j^{tr}$  in  $O^{tr}$  do {// update transient list}
  MakePrediction(KalmanFilter)
  if FindMatch( $D_t$ ) then
    DeleteSimilarObjects( $D_t$ )
    Update(KalmanFilter)
    AccumulateConfidence()
    if confidence level > threshold  $h_2$  then
      MoveObject( $O_j^{tr}, O^c$ )
      AddTemplate()
    end if
  else
    DiminishConfidence()
    if confidence < threshold  $h_3$  then
      DeleteObject( $O_j^{tr}$ )
    end if
  end if
end for
if NotEmpty( $D_t$ ) then {// deals with new detections}
  AddObject( $O^{tr}$ )
end if
```

between the optical axis and the ground plane within an upper bound of 2° and thus a reasonable assumption. The height of the camera to the ground plane P_Y as shown in Figure 11 is estimated when the camera is calibrated. The relative distance from the vehicle to the camera on the X -axis and the Z -axis can then be estimated by back projecting the touch point (where the rear end of

the vehicle touches the ground plane in the image, an example is shown in Figure 9) to the ground plane. Let $p = (p_x, p_y)$ be the touch point in the image, then the relative distance P_X and P_Z can be estimated by

$$P_Z = \frac{P_Y}{p_y - o_y} f_y, \quad P_X = \frac{f_y p_x - o_x}{f_x p_y - o_y} P_Y \quad (9)$$

Since the vehicle tracking task outputs the position and size of the vehicle, where the bounding box for the vehicle is a circle, the touch point is then defined as the bottom point on the bounding box. The relative speed V_Z can then be estimated by the numerical derivative of P_Z . To reduce the effect of noise, a FIR low-pass filter is applied on V_Z as well as smooth the results.

Vehicle/Lane Grouping

Once we have the touch point of each vehicle and the lane parameters estimated in the image, the vehicle is assigned to the lane where the touch point lies, i.e. find the nearest lane boundaries on the left and on the right in the image plane and assign the vehicle to the corresponding lane according to the two boundaries.

Fusion Module

Although we try to allocate the work into vision module and fusion module respectively, the two modules are by no means separated. On the contrary, the two modules are interlaced with each other such that a lot of work inside the vision module actually used data fusion methods. Since it is nonbeneficial to repeat works presented in the vision module, we will continue with yet-to-present work about fusion.

Combining the estimated relative speed from the vision module and our own traveling speed from the vehicle CAN bus, we can further estimate the traveling speed of each detected vehicle. Let the own traveling speed read from the CAN bus be V_0 , the relative speed of the vehicle of interest to our own vehicle is V_Z , then the traveling speed of that vehicle V_{trv} is simply

$$V_{trv} = V_0 + V_Z. \quad (10)$$

Reasoning Module

The official definition of the traffic status for the purpose of traffic control is the Level Of Service (LOS) [10] which are defined on the traffic density, which in our case is not an easy-to-be-estimated physical quantity because our own vehicle is supposed to be moving which makes the traffic density estimation at a fixed point on the road impossible. Instead, we define our own traffic status as in Table 1.

To achieve lane level traffic status reasoning, we propose to use Hidden Markov Model (HMM) based probabilistic inference technique. Here we assume lane traffics are independent. The status

Table 1: Definition of traffic status.

Green	Free flow traffic
Yellow	In between free flow and congestion
Red	Congestion or near congestion

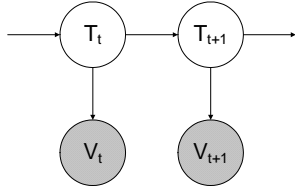


Figure 12: A graph of the HMM.

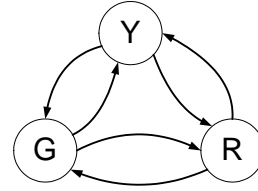


Figure 13: The state transition graph of HMM.

of the traffic is defined as $T \in \{G, Y, R\}$. The estimated speed V are quantized into n_s levels, i.e. $V \in \{v_1, \dots, v_{n_s}\}$. A graph of the HMM are shown in Figure 12, and the state transition are shown in Figure 13.

To increase the accuracy of reasoning, if nearby smart nodes are found through the communication module, those data processed by the nearby smart nodes will be collected and combined as the observation. As shown in Figure 12, T_t is the traffic status at time t and V_t is the observation dataset, which could include different number of observations based on how many smart nodes are within the communication range. The state transition probabilities and the output probabilities are estimated from the training data. Interested readers can refer to [11] for a detailed explanation of training and inference on HMM.

EXPERIMENTAL RESULTS AND CONCLUSION

Figure 15 shows a frame of the output video with vehicles tracked, relative speeds estimated and the traffic is grayscale coded in the bar on the top area of each detected lane. The numbers shown are the relative speed in the unit of MPH.

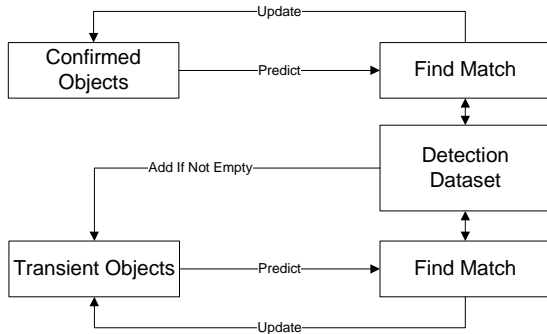


Figure 14: A flow chart of the proposed tracking algorithm.



Figure 15: A frame of the output video.

In this paper, we have presented our framework for intelligent traffic interpretation. With the help of a video camera, the vehicle CAN bus, a GPS unit and the digital map database, our system can interpret the traffic with a single node, i.e. the smart node itself, or aggregating information from nearby smart nodes to do joint inference. Possible future work includes relax the assumption that each lane are independent and infer the traffic status considering the interaction between adjacent lanes.

References

- [1] Audi of America, http://www.audiusa.com/audi/us/en2/Company/News/reducing_emissions.html.
- [2] H. Aghajan and C. Wu, "From distributed vision networks to human behavior interpretation", Behaviour Monitoring and Interpretation Workshop at the 30th German Conference on Artificial Intelligence 2007.
- [3] J. Augusto and P. McMullagh, "Ambient intelligence: concepts and applications", Int'l Journal on Computer Science and Information Systems, 4(1), pp. 1-28, 2007.
- [4] C. Fernandez et al., "Natural language descriptions of human behavior from video sequences", In 30th Annual German Conference on Artificial Intelligence (KI-2007), 2007.
- [5] K. Terzic et al., "Division of work during behavior recognition - the SCENIC approach", Workshop on Behaviour Modelling and Interpretation, 30th German Conference on Artificial Intelligence, 2007.
- [6] P. Viola and M. Jones, "Robust Real-time Object Detection", Int'l Journal of Computer Vision, 2002.
- [7] R. Lienhart and J. Maydt. "An Extended Set of Haar-like Features for Rapid Object Detection". IEEE ICIP 2002, Vol. 1, pp. 900-903, Sep. 2002.
- [8] OpenCV Documentation.
- [9] M. Fischler and R. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography", Comm. of the ACM 24: 381-395, 1981.
- [10] F. Mannering, W. Kilareski, S. Washburn, "Principles of Highway Engineering and Traffic Analysis", John Wiley & Sons; 3rd edition, pp. 170-219, 2004.
- [11] O. Cappe, E. Moulines, T. Ryddn, "Inference in Hidden Markov Models", Springer, 2005.
- [12] M. Nieto, L. Salgado, "Real-Time Vanishing Point Estimation in Road Sequences Using Adaptive Steerable Filter Banks", ACIVS07.